2011

SAH KD-Tree Construction on GPU

Zhefeng Wu，Fukai Zhao，Xinguo Liu

CAD&CG，Zhejiang University

Outline

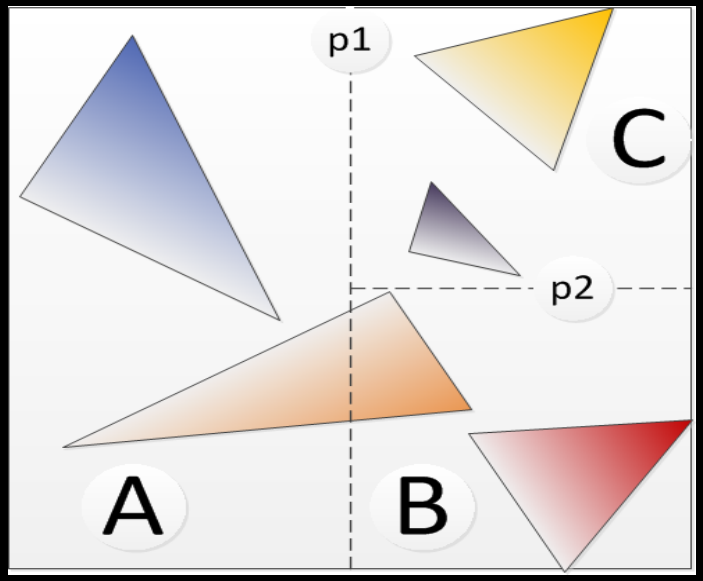
- Motivation
- Background and related work
- SAH KD-tree construction
  - *O(N log N) sequential algorithm*
  - *Parallel algorithm on GPU*
- Result and conclusion

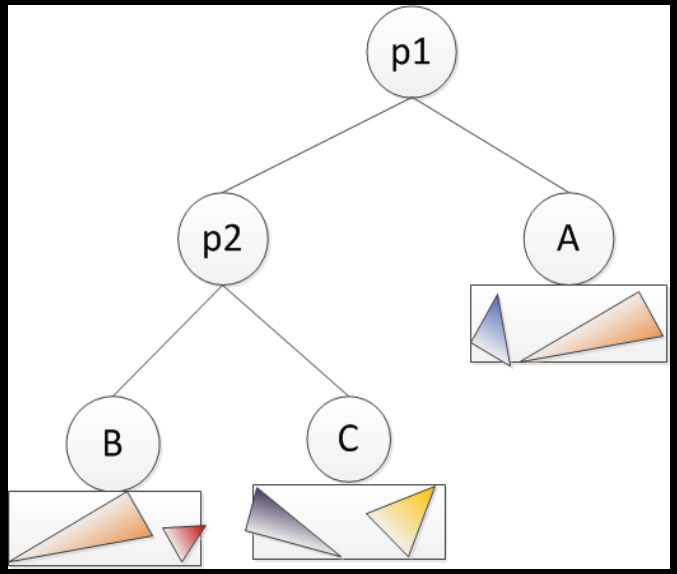
Motivation

- Ray-tracing
  - *Ray-primitive intersection*
  - *Multi-level/bounce ray-tracing*
- Render dynamic scenes
  - *Save the expensive building cost*
  - *For real-time ray-tracing*
- Goal
  - *GPU generator - speed*
  - *Precise SAH with clipping - quality*

Background and related work

- **Inner Nodes**: determine the spatial splitting
- **Leaf Nodes** : represent the primitive set





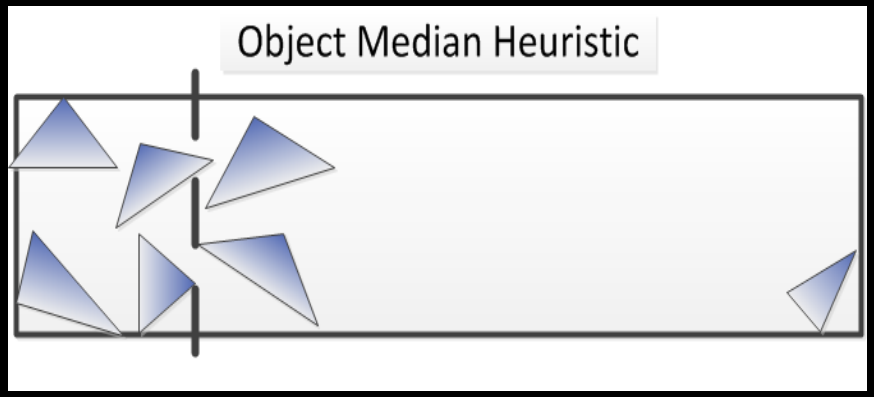
Background and related work

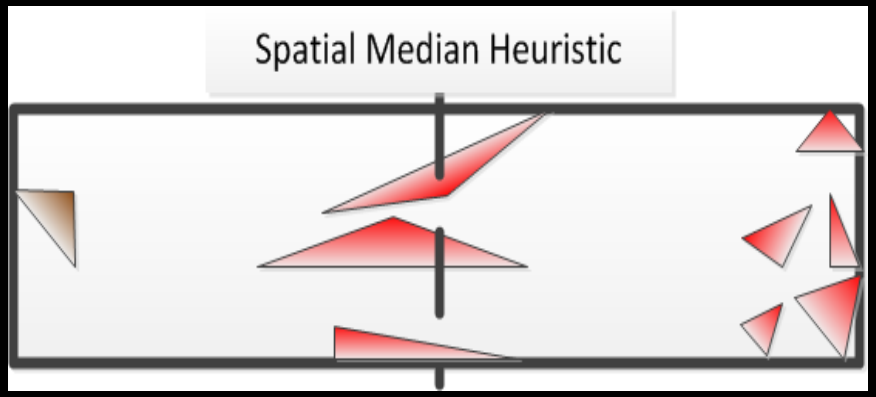
- Choose the candidate split planes
- Evaluate SAH at the candidates
- Split the node into two child nodes by the optimal split plane (with the lowest SAH)
- Distribute the primitives among children
- Repeat recursively on the children

Key Issue: how to find the best split planes

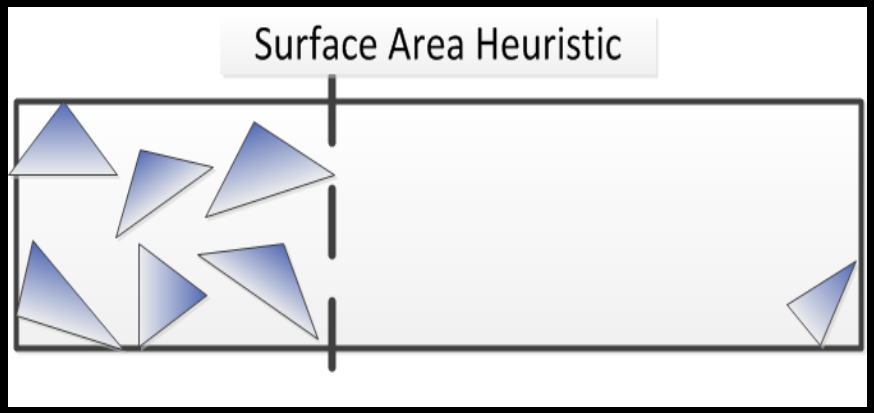
Optimal KD-tree

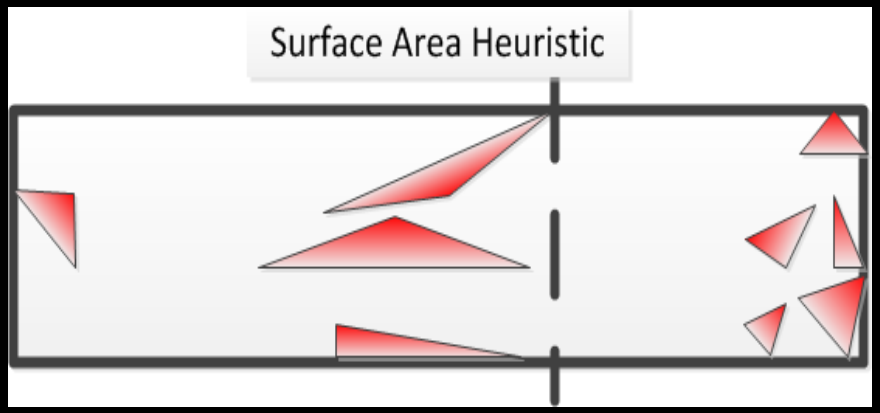
- Heuristics for partitioning





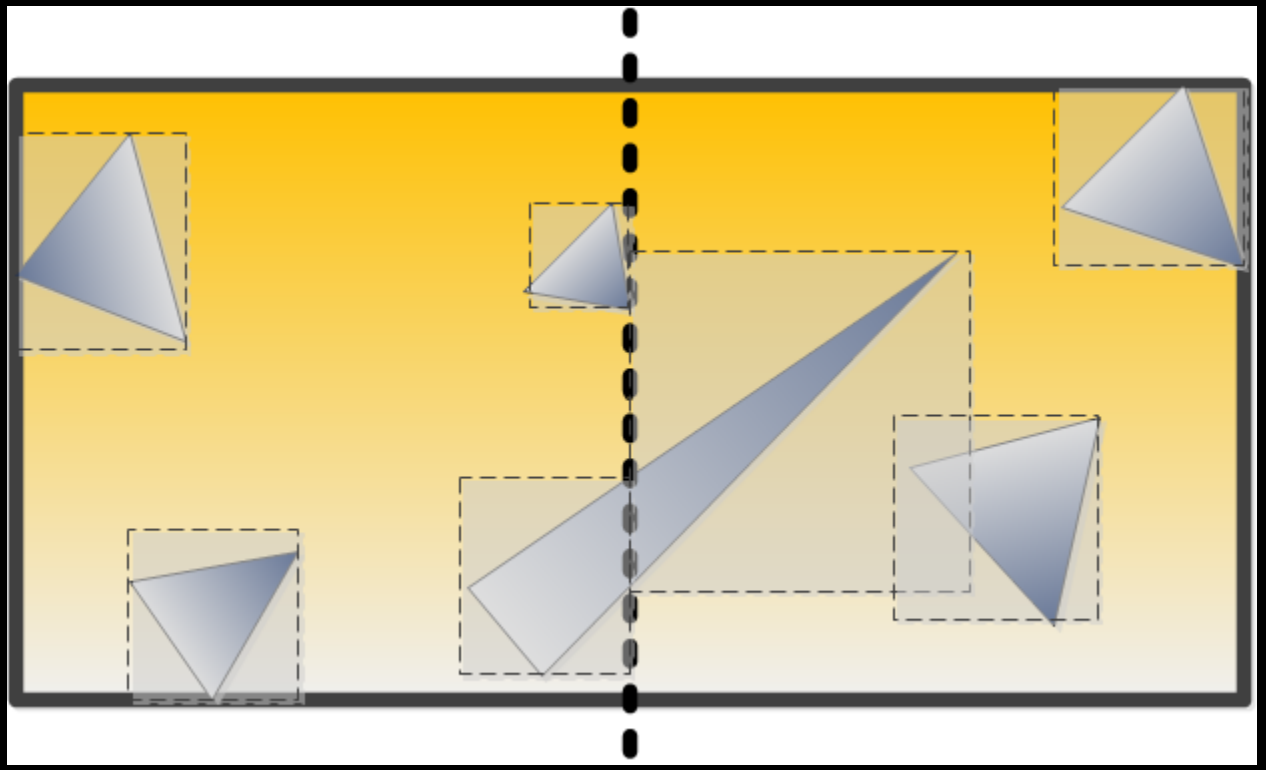
- SAH: $C_T + C_I (N_L S_L + N_R S_R) / S$





Optimal KD-tree

- Clipping the primitives against the child nodes and compress the AABBs



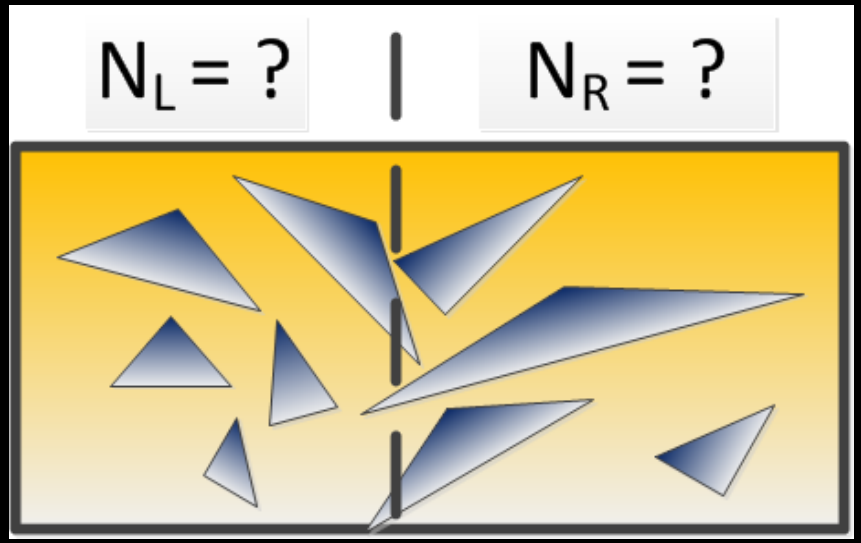
Challenges issues with SAH

- Slow to build
- Compute SAH for all candidate planes

$$C_T + C_I (N_L S_L + N_R S_R) / S$$

  - *Count the primitive numbers in both child nodes*

$$N_L \text{ and } N_R$$



Challenges issues with SAH

2011

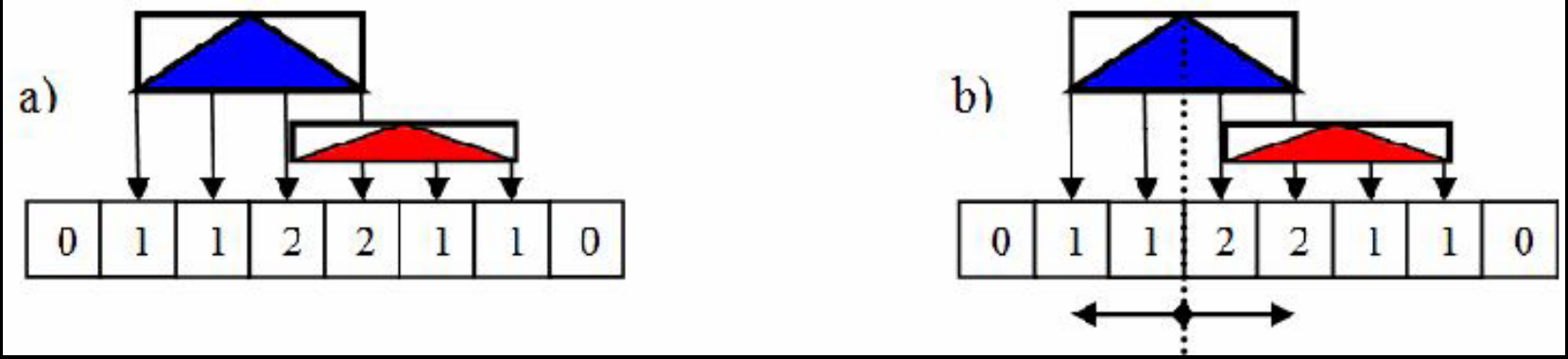
- Complexity of SAH KD-tree [Wald 2006]
  - *Naïve $O(N^2)$ method*
    - Iterating all triangles and computing N_L and N_R
  - *$O(N \log^2 N)$ method*
    - Sort the primitive AABBs in the parent node in advance
  - *$O(N \log N)$ method*
    - Reuse the order across the tree levels

the theoretical lower bound

Previous Approaches

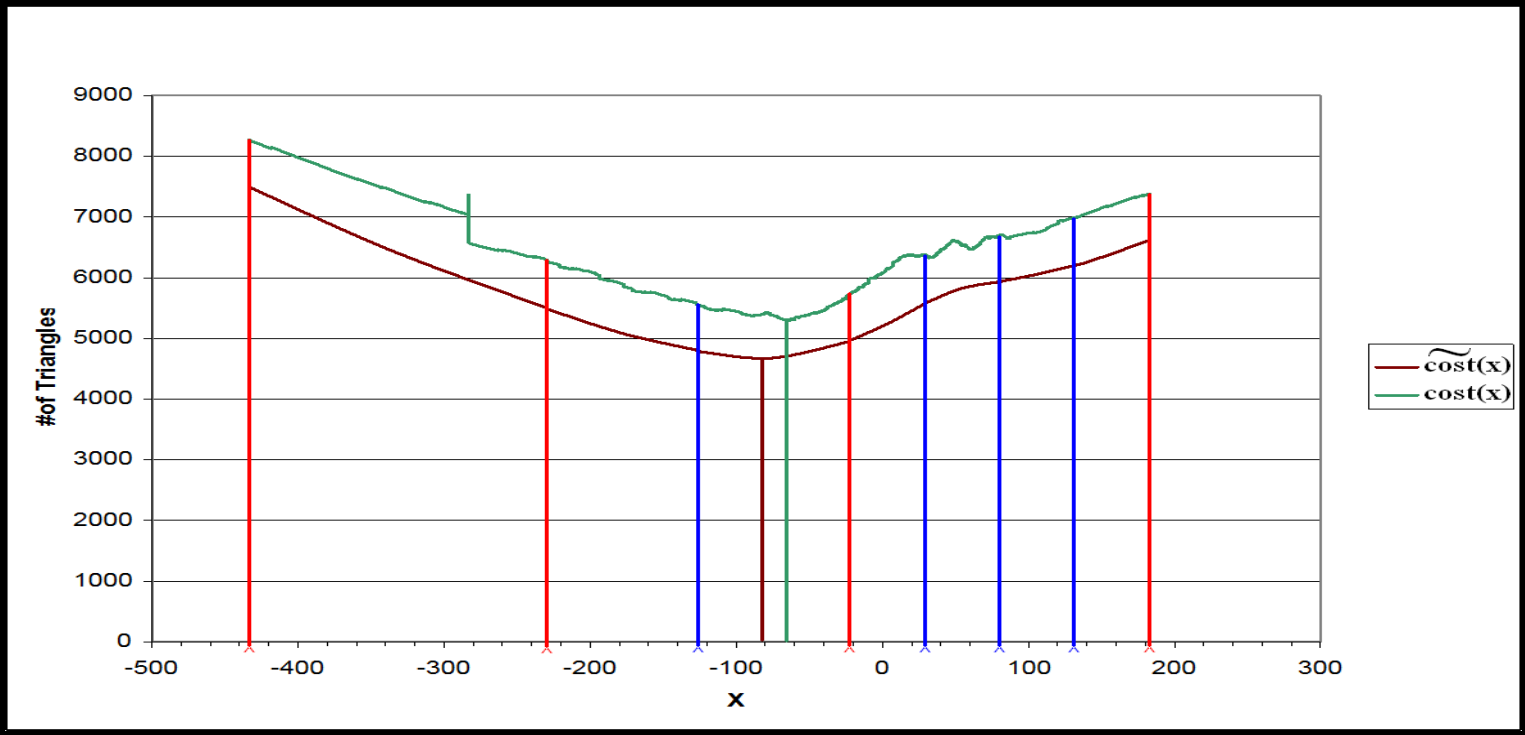
hpg 2011

- Restricting the possible split by space discretization
  - *Hurley et al. 2002, Shevtsov et al. 2007*



Previous Approaches

- Sub sampling and fitting the SAH cost function
  - *Hunt et al. 2006 (piecewise quadratic function)*



Previous Approaches

- Parallel construction on Multi-core CPUs
  - *Popov et al. 2006* *4 CPUs*
  - *Shevtsov et al. 2007* *dual core 2 CPUs*
  - *Choi et al. 2010* *32-core CPUs*

- Parallel on GPU
  - *Zhou et al. 2008*
    - spatial median split for large nodes in the upper levels
    - Switch to SAH for the small nodes in the lower levels.

Previous Approaches

	SAH Optimal Splitting				Parallel Granularity			Hardware	
	Sampling	Full	hybrid	Clip-Triangle	Subtree	Node	Triangle	CPUs	GPU
Hunt et al. [2006]	√			√				√	
Popov et al. [2006]	√			√	√			√	
Shevtsov et al [2007]	√			√	√			√	
Zhou et al [2008]			√			√	√		√

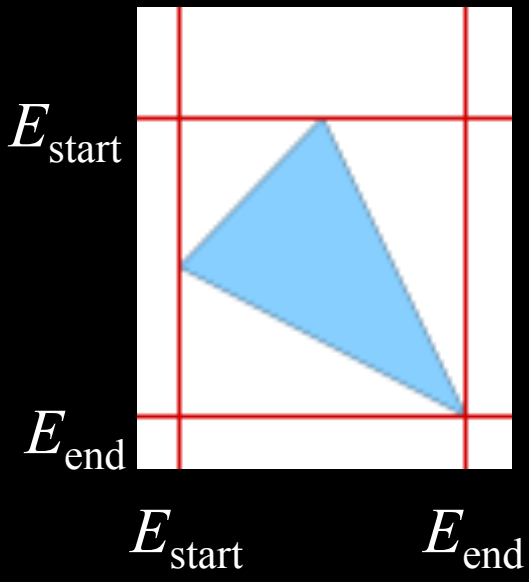
SAH KD-tree Construction

1. Choose the candidate split planes — Primitive AABBs
2. Evaluate SAH at the candidates
3. Split the node into two child nodes by the optimal split plane (with the lowest SAH)
4. Distribute the primitives among the children
5. Repeat 1~4 recursively on the children

O(N logN) Sequential Algorithm

2011

- Define events
  - *Start event – the minimum of an AABB*
  - *End event – the maximum of an AABB*



- 3 event lists in total
  - *Corresponds to the X, Y, Z axes*

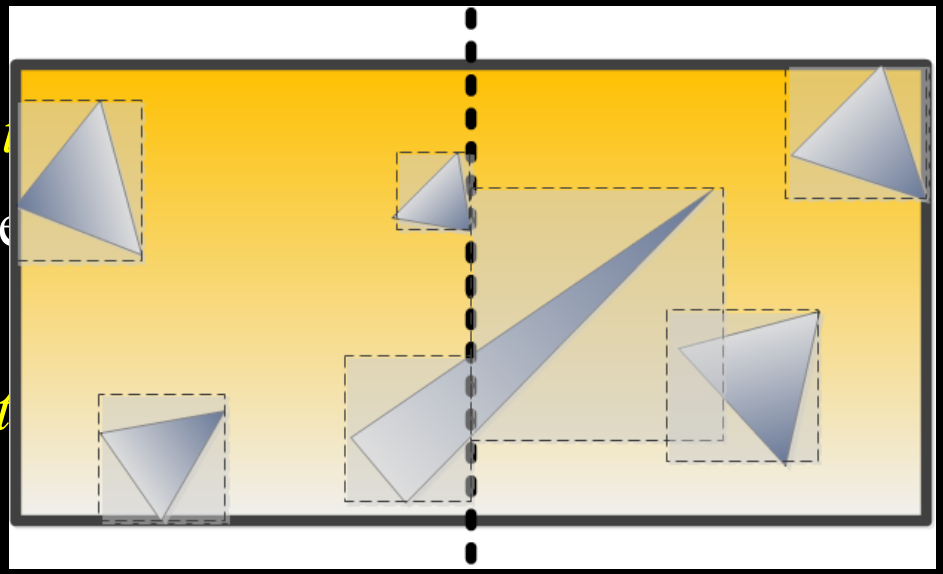
- Sort the event lists during initialization

  X: E_1 E_2, …, E_{2N}

  Y: E_1 E_2, …, E_{2N}

  Z: E_1 E_2, …, E_{2N}

O(N logN) Sequential Algorithm

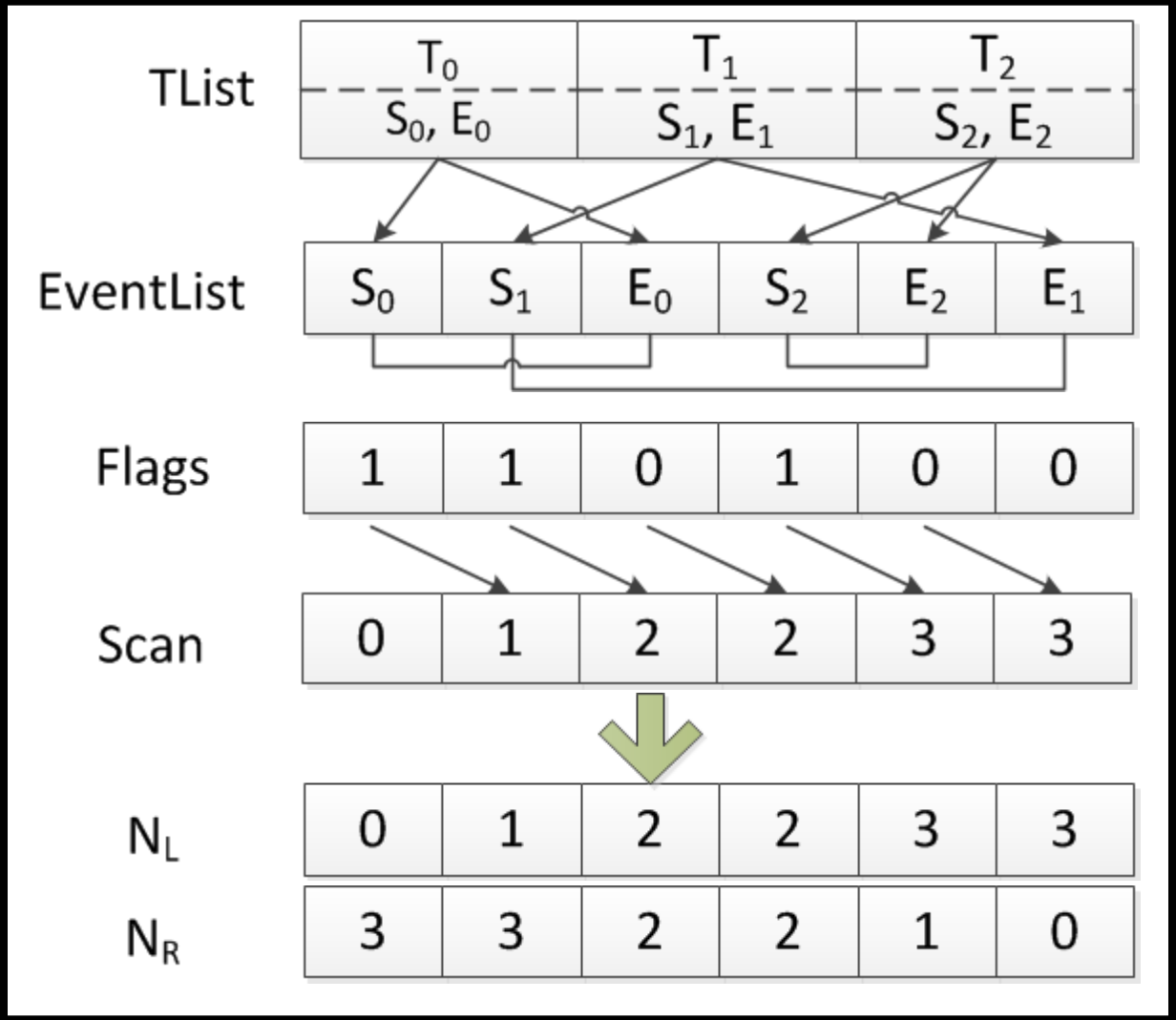


- **SelectBestPlane**
  - *Scan the sorted event lis*
    - Increase N_L for start e
    - Decrease N_R for end
  - *Evaluating the SAH for t*
- **"DivideNode"**
  - *Scan the triangles and distribute them to children*
  - *Clip the triangles against the child node's AABB*
  - *Invalidate the order of the event lists*
- **"SortNodeEvent"**
  - *Merge sort the events in each child node*

Parallel Construction on GPU

- Parallel over the triangles
  - *The same as* Lauterbach et al. did [2009]
  - Different from [*Zhou et al. 2008] over nodes*
- Using standard parallel scan primitives to compute N_L and N_R ?

Parallel Construction on GPU hpg 2011



N_T: No. of tris

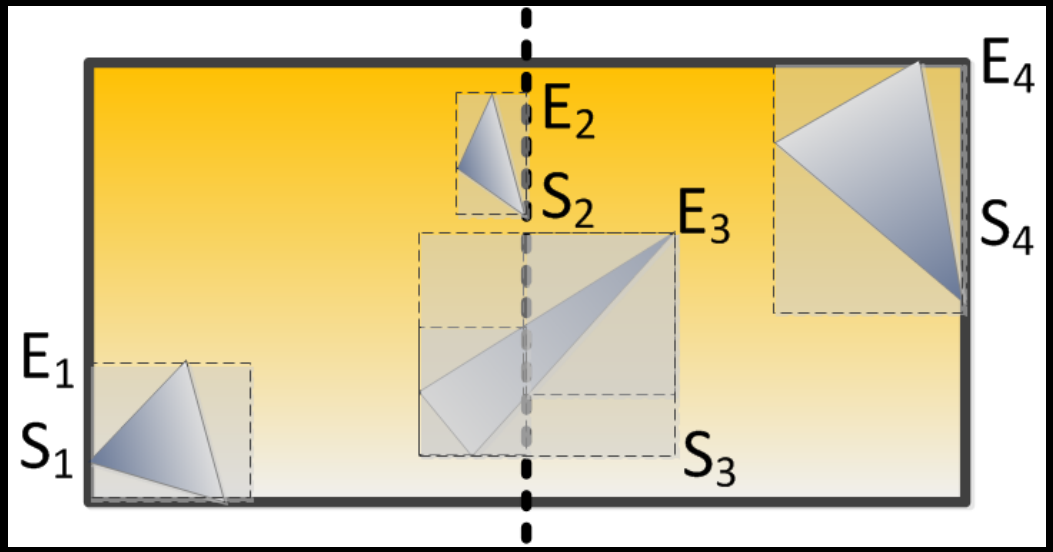
k-Scan[k]: No. of end events

$N_L[k] = scan[k]$

$N_R[k] = N_T - (k - Scan[k]) - (1 - Flags[k])$

Parallel Construction on GPU hpg 2011

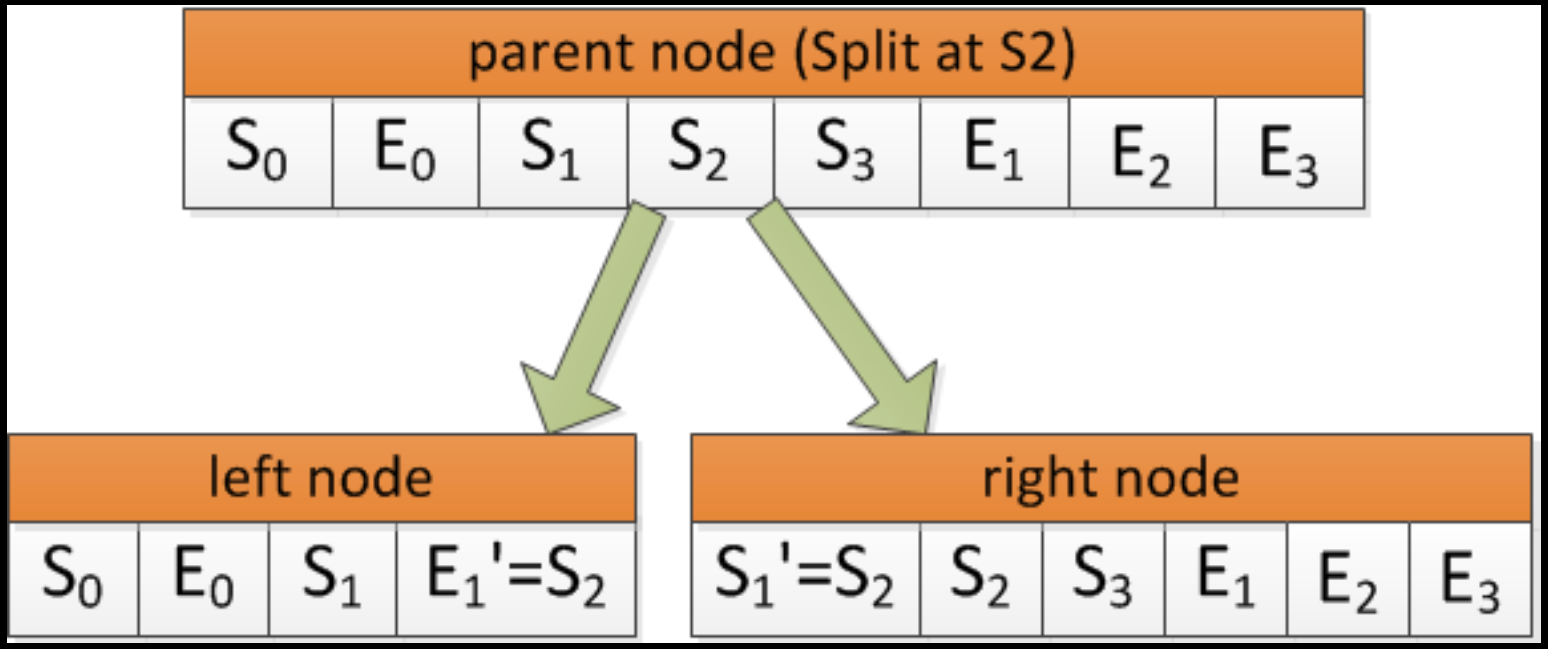
- Clipping the triangles against the children
  - *Invalidates the ordered event lists*



Observation: Most are lined in order except those new events generated by clipping. $\sim N^{1/2}$

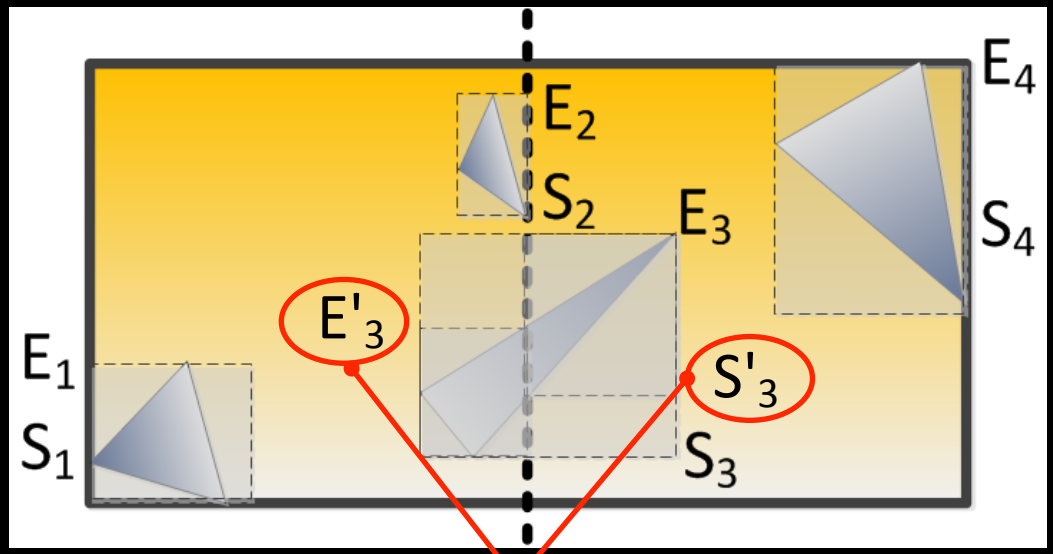
Parallel Construction on GPU

- Case I: on the splitting axis
  - *Order is inherited*

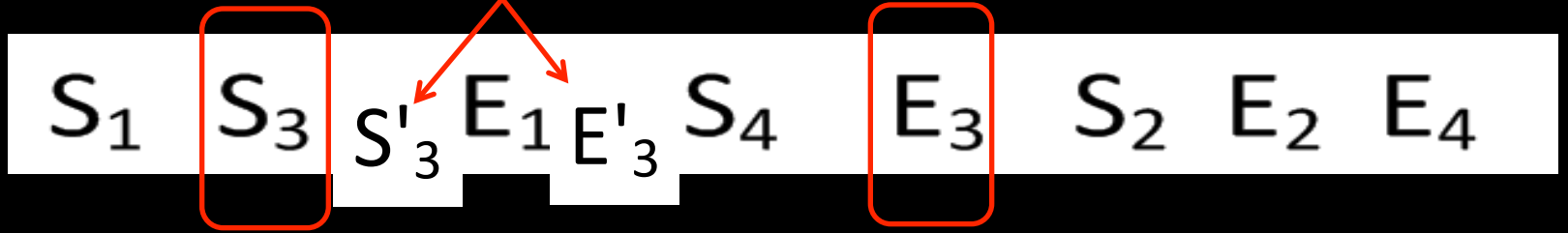


Parallel Construction on GPU hpg 2011

- Case II: ***other than*** the splitting axis
  - *The order is almost inherited except for a small part…*



$\sim N^{1/2}$



Parallel Bucket-based Sorting

2011

Let $E_{parent} = E_1, E_2, \ldots E_{2M}$ (ordered)

$E_{child} = e_1, e_2, \ldots e_{2N}$ (almost ordered, except…)

Bucket set = $[E_1, E_2) \cup [E_2, E_3) \cup \ldots \cup [E_{2M-1}, E_{2M}]$

- Using Buckets to sort E_{child}

1. Find the bucket (interval) that contains $e_{i,}$ the event of the child nodes
2. Get the order index by brute-force comparison inside the intervals.

Results

hpg 2011

- GTX280 with 1GB memory
- Intel Xeon dual-core 3.0G CPU with 4GB main memory
- Stack-based tracer [Pharr and Humphreys 2004]

Results

- Demo1
- Demo2

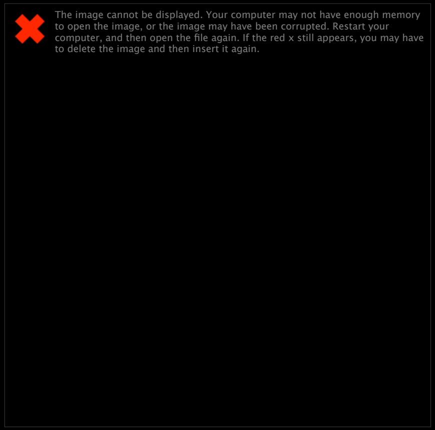
V.S. multi-core CPU method

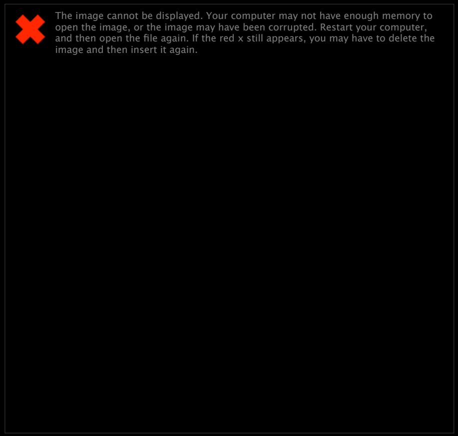
1024x1024 resolution		CPUs SAH KD-Tree		GPU SAH KD-Tree	
Scene	Triangle	Build	Trace	Build	Trace
Bunny	69K	0.068s	n/a	0.059s	0.031s
Angel	474K	0.337s	n/a	0.311s	0.036s
Dragon	871K	0.654s	n/a	0.511s	0.041s
Happy	1087K	0.835s	n/a	0.645s	0.051s

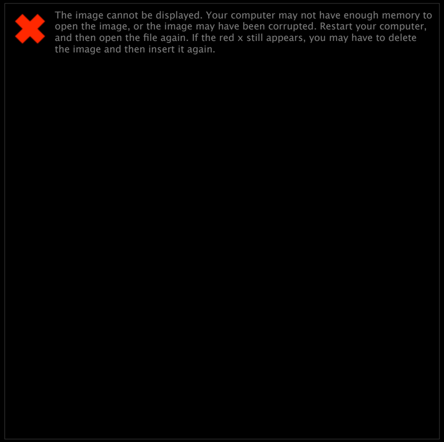
32-cores CPU, cache-coherent, shared-memory machine [Choi et al. 2010], Full SAH KD-Tree without triangle clipping

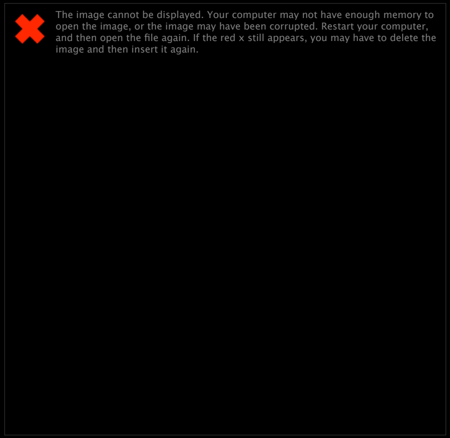
V.S. SAH BVH-Tree

hpg 2011





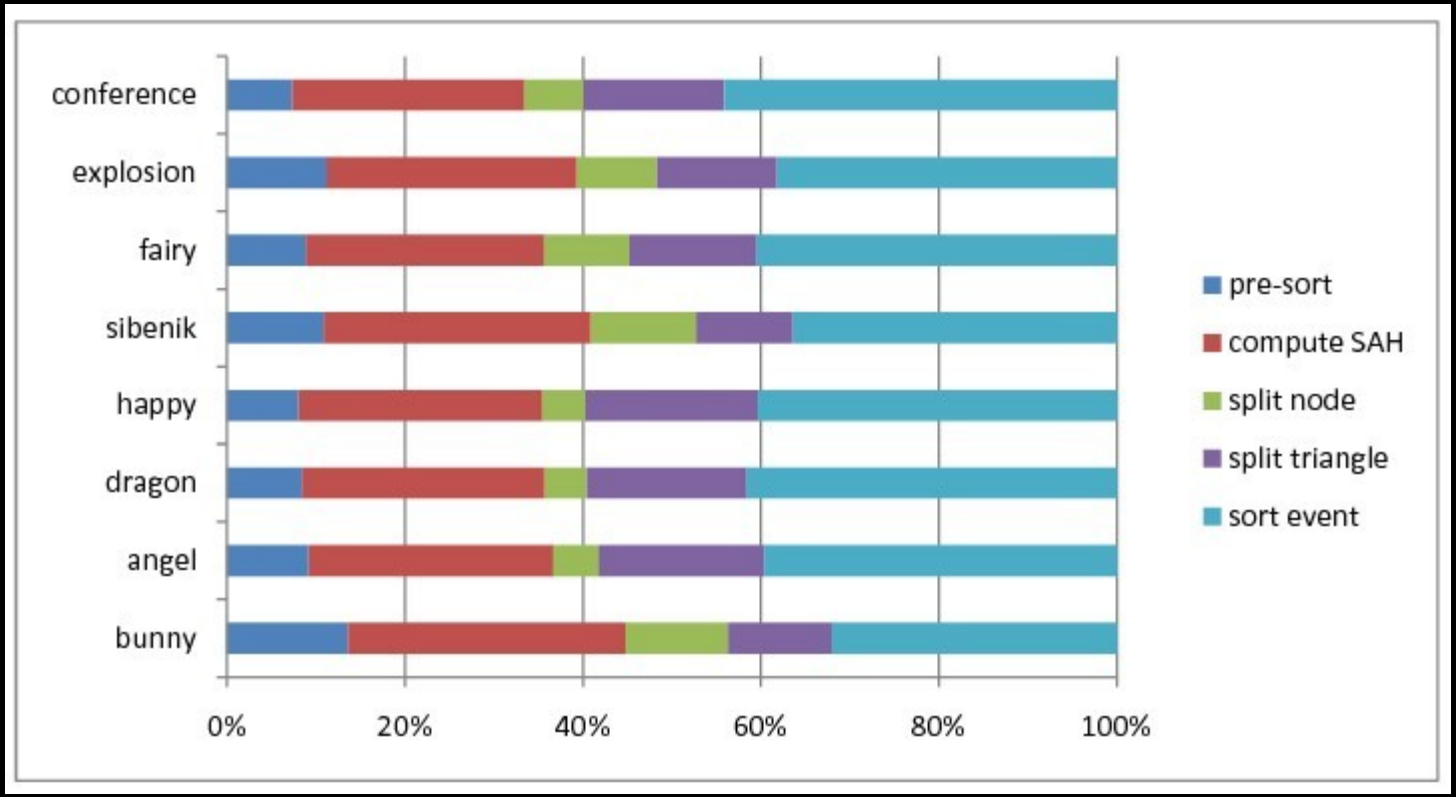




1024x1024 resolution		GPU SAH BVH-Tree		GPU SAH KD-Tree	
Scene	Triangle	Build	Trace	Build	Trace
Sibenik	82K	0.144s	21.7fps	0.091s	24.4fps
Fairy	174K	0.488s	21.7fps	0.142s	31.2fps
Explosion	252K	0.403s	7.75fps	0.161s	32.1fps
Conference	284K	0.477s	24.5fps	0.258s	32.2fps

GTX280 with 1GB memory [Lauterbach et al. 2009]

Stage Time Analysis



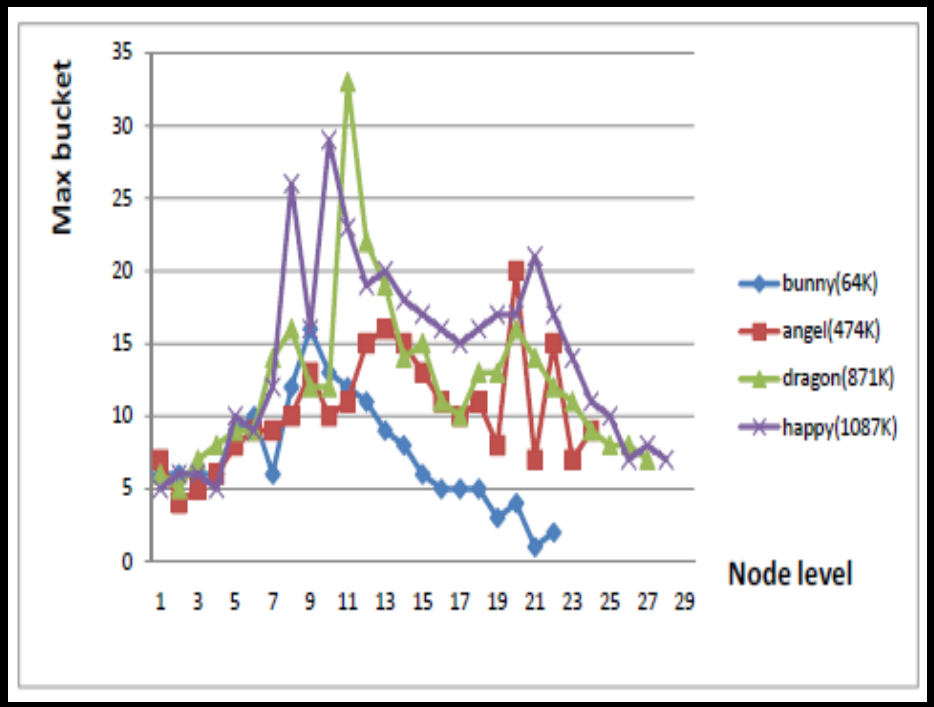
"sort event" is about 1.5 times of "compute SAH"

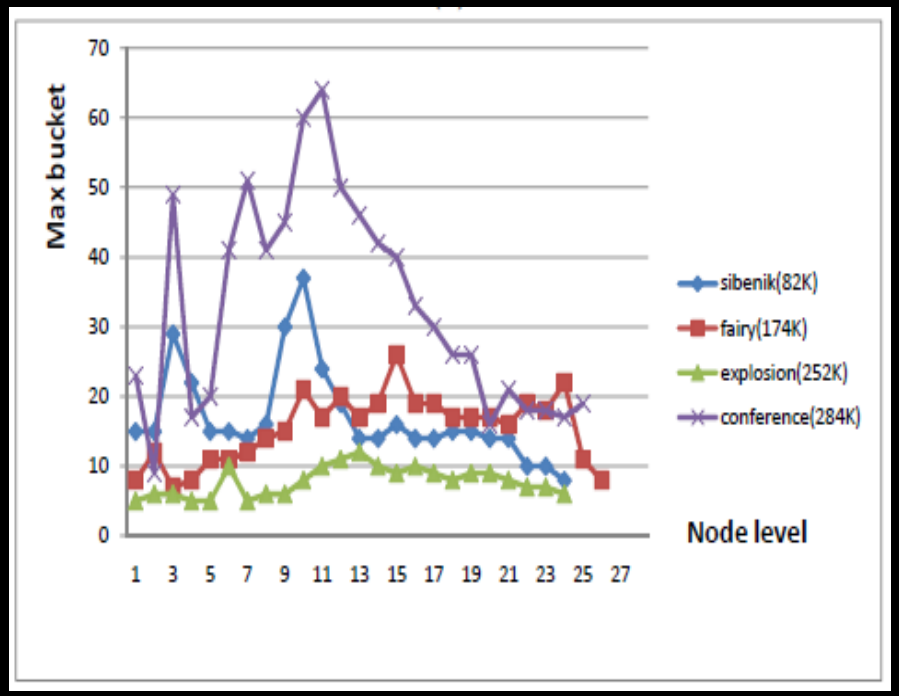
Memory Analysis

Scene	Triangles	Peak-Memory	Final-Memory
Bunny	69K	33.96MB	4.86MB
Sibenik	82K	39.34MB	7.71MB
Fairy	174K	80.33MB	14.91MB
Explosion	252K	86.48MB	16.36MB
Conference	284K	159.58MB	28.74MB
Angel	474K	218.26MB	34.33MB
Dragon	871K	417.33MB	69.76MB
Happy	1087K	512.65MB	87.08MB

peak memory is about 5 ~7 times of the kd-tree storage

Bucket-based Sort Analysis





The greatest maximum size appears at the middle level

Conclusion

- A GPU KD-tree generator
  - *Precise SAH at all levels*
  - *Clipping triangles*
  - *Parallel on primitives*
- A bucket-based sort algorithm for the event list
- Limitations
  - *High memory consumption*
  - *Handle triangles for now*

2011

Thanks for your attention

Questions ?

{wuzhefeng, xgliu} @ cad.zju.edu.cn